Міністерство освіти і науки України

Національний Технічний Університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

Криптографія

Комп’ютерний практикум №4

Вивчення криптосистеми RSA та алгоритму електронного підпису; ознайомлення з методами генерації параметрів для асиметричних криптосистем

Виконали:

Студент групи ФБ-84

Чипчев Дмитро

Студент групи ФБ-84

Киричук Тарас

Перевірив:

Чорний О.М.

Київ - 2020

Мета та основні завдання роботи

Ознайомлення з тестами перевірки чисел на простоту і методами генерації ключів для асиметричної криптосистеми типу RSA; практичне ознайомлення з системою захисту інформації на основі криптосхеми RSA, організація з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів.

Порядок і рекомендації щодо виконання роботи:

1. Написати функцію пошуку випадкового простого числа з заданого інтервалу або заданої довжини, використовуючи датчик випадкових чисел та тести перевірки на простоту. В якості датчика випадкових чисел використовуйте вбудований генератор псевдовипадкових чисел вашої мови програмування. В якості тесту перевірки на простоту рекомендовано використовувати тест Міллера-Рабіна із попередніми пробними діленнями. Тести необхідно реалізовувати власноруч, використання готових реалізацій тестів не дозволяється.

2. За допомогою цієї функції згенерувати дві пари простих чисел p, q і 1 1 p , q довжини щонайменше 256 біт. При цьому пари чисел беруться так, щоб $pq \leq p1q1$; p і q – прості числа для побудови ключів абонента A, 1 p і q1 – абонента B.

3. Написати функцію генерації ключових пар для RSA. Після генерування функція повинна повертати та/або зберігати секретний ключ (d, p,q) та відкритий ключ (n,e) . За допомогою цієї функції побудувати схеми RSA для абонентів A і B – тобто, створити та зберегти для подальшого використання відкриті ключі (e,n) , (,) 1 n1 e та секретні d і d1 .

4. Написати програму шифрування, розшифрування і створення повідомлення з цифровим підписом для абонентів A і B. Кожна з операцій (шифрування, розшифрування, створення цифрового підпису, перевірка цифрового підпису) повинна бути реалізована окремою процедурою, на вхід до якої повинні подаватись лише ті ключові дані, які необхідні для її виконання. За допомогою датчика випадкових чисел вибрати відкрите повідомлення M і знайти криптограму для абонентів A и B, перевірити правильність розшифрування. Скласти для A і B повідомлення з цифровим підписом і перевірити його.

5. За допомогою раніше написаних на попередніх етапах програм організувати роботу протоколу конфіденційного розсилання ключів з підтвердженням справжності по відкритому каналу за допомогою алгоритму RSA. Протоколи роботи кожного учасника (відправника та приймаючого) повинні бути реалізовані у вигляді окремих процедур, на вхід до яких повинні подаватись лише ті ключові дані, які необхідні для виконання. Перевірити роботу програм для випадково обраного ключа $0 < k < n$. Кожна з наведених операцій повинна бути реалізована у вигляді окремої процедури, інтерфейс якої повинен приймати лише ті дані, які необхідні для її роботи; наприклад, функція Encrypt(), яка шифрує повідомлення для абонента, повинна приймати на вхід повідомлення та відкритий ключ адресата (і тільки його), повертаючи в якості результату шифротекст. Відповідно, програмний код повинен містити сім високорівневих процедур: GenerateKeyPair(), Encrypt(), Decrypt(), Sign(), Verify(), SendKey(), ReceiveKey(). Кожну операцію рекомендується перевіряти шляхом взаємодії із тестовим середовищем, розташованим за адресою http://asymcryptwebservice.appspot.com/?section=rsa. Наприклад, для перевірки коректності операції шифрування необхідно а) зашифрувати власною реалізацією повідомлення для серверу та розшифрувати його на сервері, б) зашифрувати на сервері повідомлення для вашої реалізації та розшифрувати його локально.

Хід роботи та опис труднощів

Спочатку було створено функцію для перевірки числа на простоту (Міллера-Рабіна).Після цього було сформовано генератором випадкових чисел дві пари простих чисел p , q і p_1 ,q_1 (256 біт) , такі що : pq <= p_1q_1 , після цього було обраховано функцію Ойлера для чисел p , q , далі генератором випадковим чисел було обрано число e , та за допомогою функції reverse() було обраховано d.Після всіх цих перетворень було отримано відкритий (n,e) та секретний ключ (d,p,q). Було реалізовано функції : GenerateKeyPair(), Encrypt(), Decrypt(), Sign(), Verify(), SendKey(), ReceiveKey().

Вибрані числа для A і B :

p =
57931198033881302057475356366846202577190027935161588533895519057053608224863

q =
499433011172986904479497465703992414848946220974139765373698039832058858739979

n = p*q
=289327526749199573238352884774457039276627293489644429109633388248451862183840008191429671885162535581535071784712780460703639481204022912336038151253987 7

e
=216513781887143291455298847639519401069822871316478438866151053696428909674110403865339725889359902143567465397556682522710843925144229282761937815808718 5

d =
126809981579892547387247642273775044261541116541758652654705154078122600167431523202074424824320523125586268362805387304327686193891243520108441432463578 5

p1=
905008873742502286321856932969588870653723725816566837252600104067163820832 07

q1=
500179316983205085068735346776618751786139948680685411591389692028112399915837
n1=p1*q1=
452666720332264480396859883890970564700402288445634447795169823702845227463903757506867295014703075429543523458156459459798737174862971633152398141104925 9
e1 =
240878323129257962918638790209570031548121138573349001088767754106033428125500211053020107064734598884643260843495149568709004320936144333795037875683336 3
d1=
169104794722963106780881160282219594280076161474529170844520060425705736474784233596440308060948077988760836710734794805624921392457664547554853456498979

Чисельні значення прикладів ВТ, ШТ, цифрового підпису:

ВТ:
828665984348922309492550446789507359868008099512197710045977360821798732321992024928980413149912150212242334741207458310716685107310652898669535202117115

ШТ:
157386598942367770206198200315138349173433346532383970031316453174286513472133215253485486632288993406247628773655124205153908267393280417409593884874673

Цифровий підпис:
133552060328135807751401804210280471795391011342209323895954326803485236567297762837521260337658212596391691006129016383580512042741425146922263658522107 8

Опис кроків протоколу конфіденційного розсилання ключів з підтвердженням справжності, чисельні значення характеристик на кожному кроці:

Відкритий ключ абонента А(відправник) : (e,n)

e=0x363d3543572f3f273d774d4ee609b64be549d5ce5919370eeedf6febbbb865ad83a9909854ee3d6df59096cace7d877f7bab8e517e052586d4e22840045376d

n=0x2eebb3683214e61d12a8d64b1e16c062f643320d6703dcde543762962918694a0d7879c5e31a99958749d5b0a007e7611e3085df73cbf8599c3a153c37b83f1d

Відкритий ключ абонента В(одержувач) : (e_1,n_1)

e_1=0xcb59755b02ec11a174e9bded1ece6019bb68b43d4d6cbb1c420342d8f07f8a5f

n_1=0x10001

Генеруємо випадкове число k , яке $0 < k < n$:

k=0x8c7129de0737da0f3a63aaece0fd97b71a6d3f3f382f10672d9709d78d013a8f5e2052d6553c67fdb3400932e47556eb10414c2c74560608784c9d2f75a4419

Зашифровуємо за формулою $k_1 = k^{e1} \bmod n_1$, звідси:

k_1=0xb6b7870195a06ca74bd1c7ea040cc15641c60fa4a56328bbc601742c29242a31

Робимо підпис k за формулою $s = k^d \bmod n$, звідси:

s=0xf241d7c60aaf0b7285768de223718a11467baec364e66a44f98b19af9f4f3ba8630e4a625da8643a518c5c306e5cd5bb353f1786e40a251ba35883cc8067f12

Підписуємо s за формулою $s_1 = s^{e1} \bmod n_1$, звідси:

s_1=481f35fefcd4d01f8a531c821af1ab75a475510439c985e6e289d9c0d298d7f9

Висновок

В ході лабораторної роботи ми ознайомились з тестом перевірки числа на простоту (функція Міллера- Рабіна), і методами генерації ключів для асиметричної криптосистеми типу RSA. Ознайомились з системою захисту інформації на основі криптосхеми RSA, організація з використанням цієї системи засекреченого зв'язку й електронного підпису, вивчення протоколу розсилання ключів.